Homework 9: Product Search iOS App

Table of Contents

1. Objectives

- Become familiar with the Swift language, Xcode and iOS App development.
- Practice the Model-View-Controller design pattern.
- Build a good-looking iOS app.
- Learn to integrate APIs and iOS SDK
- Manage and use third-party libraries by CocoaPods

2. Background

2.1 Xcode

Xcode is an integrated development environment (IDE) containing a suite of software development tools developed by Apple for developing software for macOS and iOS. First released in 2003, the latest stable release is version 10.1 and is available via the Mac App Store free of charge.

Features:

- Swift 4.x support
- Playgrounds
- Interface Builder
- Device simulator and testing
- User Interface Testing
- Code Coverage

The Official homepage of the Xcode is located at:

https://developer.apple.com/xcode/

2.2 iOS

iOS (originally iPhone OS) is a mobile operating system created and developed by Apple Inc. and distributed exclusively for Apple hardware. It is the operating system that presently powers many of the company's mobile devices, including the iPhone, iPad, and iPod touch. It is the second most popular mobile operating system in the world by sales, after Android.

The Official iOS home page is located at:

http://www.apple.com/iOS/

The Official iOS Developer homepage is located at:

https://developer.apple.com/ios/

2.3 Swift

Swift is a general-purpose, multi-paradigm, compiled programming language created for iOS, macOS, watchOS, tvOS and Linux development by Apple Inc. Swift is designed to work with Apple's Cocoa and Cocoa Touch frameworks and the large body of existing Objective-C code written for Apple products. Swift is intended to be more resilient to erroneous code ("safer") than Objective-C and also more concise. It is built with the LLVM compiler framework included in Xcode 6 and later and uses the Objective-C runtime, which allows C, Objective-C, C++ and Swift code to run within a single program.

The Official Swift homepage is located at:

https://developer.apple.com/swift/

2.4 Amazon Web Services (AWS)

AWS is Amazon's implementation of cloud computing. Included in AWS is Amazon Elastic Compute Cloud (EC2), which delivers scalable, pay-as-you-go compute capacity in the cloud, and AWS Elastic Beanstalk, an even easier way to quickly deploy and manage applications in the AWS cloud. You simply upload your application, and Elastic Beanstalk automatically handles the deployment details of capacity provisioning, load balancing, auto-scaling, and application health monitoring. Elastic Beanstalk is built using familiar software stacks such as the Apache HTTP Server, PHP, and Python, Passenger for Ruby, IIS for .NET, and Apache Tomcat for Java.

The Amazon Web Services homepage is available at: http://aws.amazon.com/

2.5 Google App Engine (GAE)

Google App Engine applications are easy to create, easy to maintain, and easy to scale as your traffic and data storage needs change. With App Engine, there are no servers to maintain. You simply upload your application and it's ready to go. App Engine applications automatically scale based on incoming traffic. Load balancing, micro services, authorization, SQL and noSQL databases, memcache, traffic splitting, logging, search, versioning, roll out and roll backs, and security scanning are all supported natively and are highly customizable.

To learn more about GAE support for PHP visit this page:

https://cloud.google.com/appengine/docs/php/

To learn more about GAE support for Node.js visit this page:

https://cloud.google.com/appengine/docs/standard/nodejs/

3. Prerequisites

This homework requires the use of the following components:

3.1 Download and install the latest version of Xcode

To develop iOS apps using the latest technologies described in these lessons, you need a Mac computer (macOS Sierra 10.12.4 or later) running the latest version of Xcode. Xcode includes all the features you need to design, develop, and debug an app. Xcode also contains the iOS SDK, which extends Xcode to include the tools, compilers, and frameworks you need specifically for iOS development.

Download the latest version of Xcode on your Mac for free from the App Store.

To download the latest version of Xcode:

- Open the App Store app on your Mac (by default it's in the Dock).
- In the search field in the top-right corner, type Xcode and press the Return key.
- The Xcode app shows up as the first search result.
- Click Get and then click Install App.
- Enter your Apple ID and password when prompted.
- Xcode is downloaded into your Applications directory.

You may use any other IDE other than Xcode, but you will be on your own if problems come up.

Swift 4.2 is strongly recommended to use for this app.

3.2 Add your account to Xcode

When you add your Apple ID to the Xcode Accounts preferences, Xcode displays all the teams you belong to. Xcode also shows your role on the team and details about your signing identities and provisioning profiles that you'll create later in this document. If you don't belong to the Apple Developer Program, a personal team appears.

Here is detailed documentation:

https://developer.apple.com/library/iOS/documentation/IDEs/Conceptual/AppStoreDistributionTutorial/AddingYourAccounttoXcode/AddingYourAccounttoXcode.html

3.3 Install CocoaPods

CocoaPods is a dependency manager for Swift and Objective-C Cocoa projects. It has over ten thousand libraries and can help you scale your projects elegantly. You can install dependencies using it, we will need to install many third-party modules and frameworks using it.

CocoaPods is built with Ruby and is installable with the default Ruby available on macOS. We recommend you use the default Ruby. Using the default Ruby install can require you to use 'sudo' when installing gems.

Run the command below in your Mac terminal:

```
$ sudo gem install cocoapods
```

Once you have created your Xcode project, you can start to integrate CocoaPods into your project.

Further guides on how to integrate CocoaPods are available at: https://cocoapods.org/.

The following pods will be needed:

- SwiftyJSON: To handle JSON data.
- Alamofire: To make HTTP requests.

- McPicker ~> 2.0.0* : To implement the picker view in Figure 2
- Toast-Swift ~> 4.0.0* : To display messages in your app
- SwiftSpinner : For the big spinner as shown in Figure 6.

*The versions of the pods mentioned are compatible with Swift 4.2.

4. High Level Design

This homework is a mobile app version of Homework 8. In this exercise, you will develop an iOS Mobile application, which allows users to search for products using the eBay APIs, get product details, add products in wishlist, and post on Facebook. You should reuse the backend service (node.js script) you developed in HW8 and follow the same API call requirements.

The main scene of this app is like that in Figure 1. The launch screen of the App is shown in Figure 1.1. All the implementation details and requirements will be explained in the following sections.

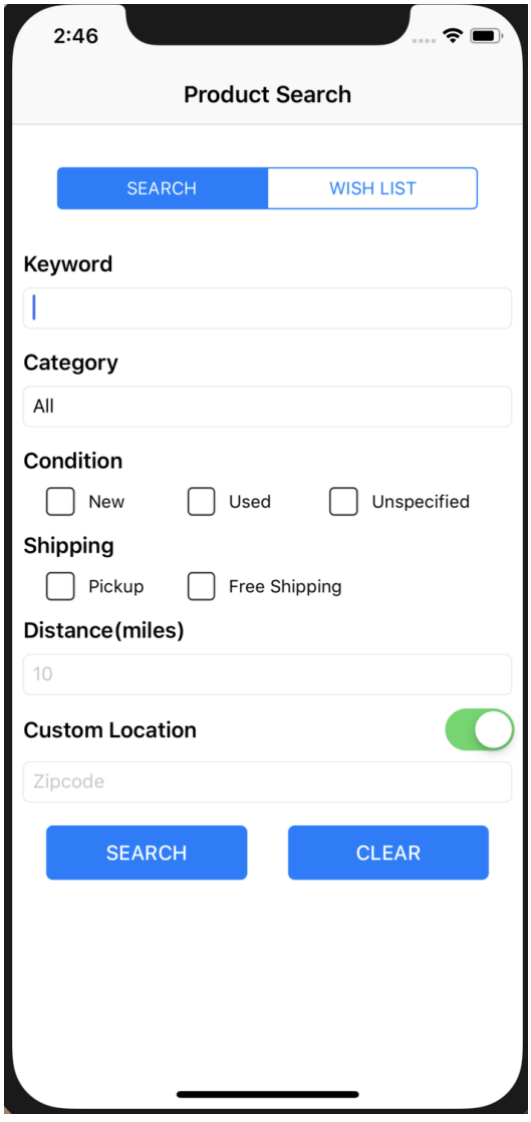


Figure 1: Product Search App

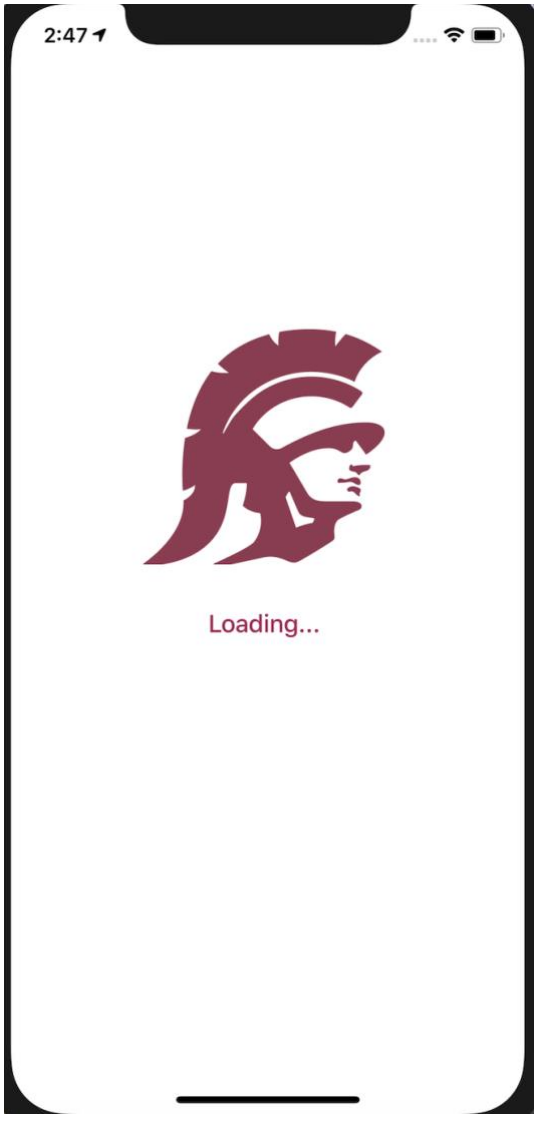


Figure 1.1: Launch Screen

5. Implementation

5.1 Search Form

You must replicate the Search Form, as shown in Figure 1.

The interface consists of the following:

- **Keyword**: A 'UITextField' component allowing the user to enter the keyword.
- **Category**: A 'UITextField' allowing the user to choose a category. When the user taps on this field, a picker view should display at the bottom of the screen for selecting a category, as shown in Figure 2. Make sure you include all the categories in homework 8.

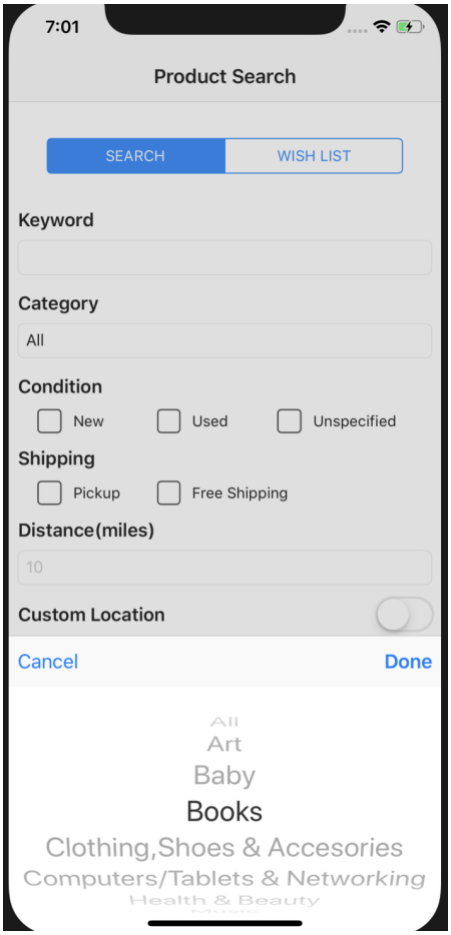


Figure 2: Choose category from a picker view

- **Condition:** Three 'UIButtons' that 'act' like checkboxes. Multiple options can be selected at once.
- **Shipping:** Two 'UIButtons' that 'act' like checkboxes. Both options can be selected at once.
- **Distance**: A 'UITextField' component allowing the user to enter the distance (in miles).
- **Custom Location**: A 'UISwitch' that toggles between current location (when turned OFF) and custom location (when turned ON) which is provided in the form of a zipcode.

- **Zipcode**: A 'UITextField' which is shown/hidden based on the state of UISwitch. It provides the autocomplete function as shown in Figure 3. Make sure you use the same API as Homework 8. (Hint: The results can be shown as a 'UITableView' which is hidden/shown according to the response of the autocomplete API).

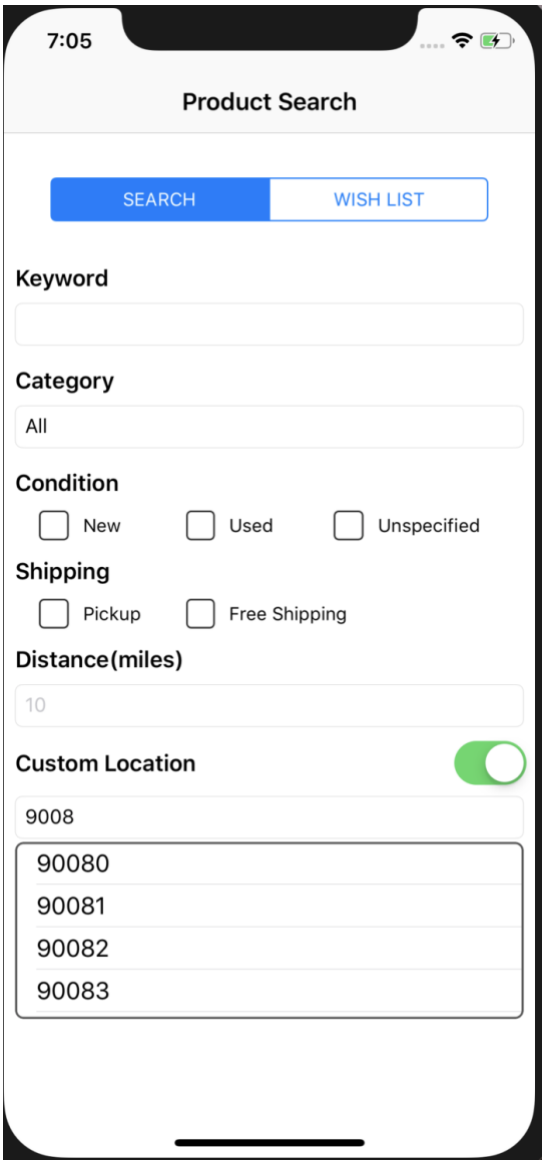


Figure 3: Autocomplete for Zip code

- A **SEARCH button** to get the input information of each field, after validation. If the validation is successful, then the products would be fetched from the server. However, if the validations are unsuccessful, appropriate messages should be displayed and no further requests would be made to the server.
- A **CLEAR button** to clear the input fields and set them to default values if applicable. It should remove all error messages (if present).

5.1.1 Current Location

When 'UISwitch' is OFF, the current location should be used. The current location can be obtained by using ip-api.com or the standard location service provided by apple.
Both ways are acceptable for this HW, but in general, Apple's location service should be used if the app requires high precision locations or more frequent updates.
More details on standard location service can be found here:

https://developer.apple.com/documentation/corelocation/getting_the_user_s_location/using_the_standard_location_service

5.1.2 Validation

The validation for empty keyword and empty Zipcode (if custom location switch is turned on) needs to be implemented. If the user does not enter anything in the 'UITextField' or just enters some empty spaces, when he presses the 'SEARCH' button you need to display an appropriate message to indicate the error, as shown in Figure 4 and 5.

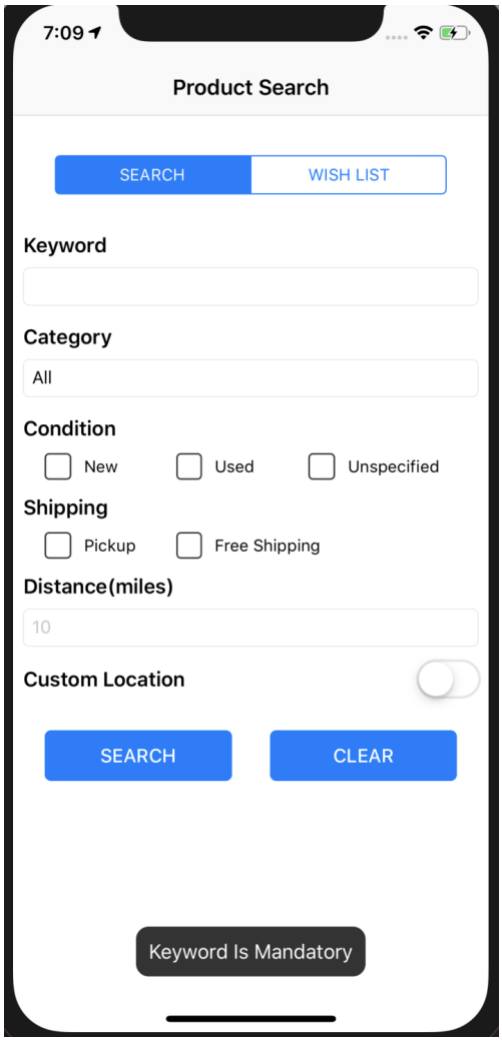


Figure 4: Validation for Keyword

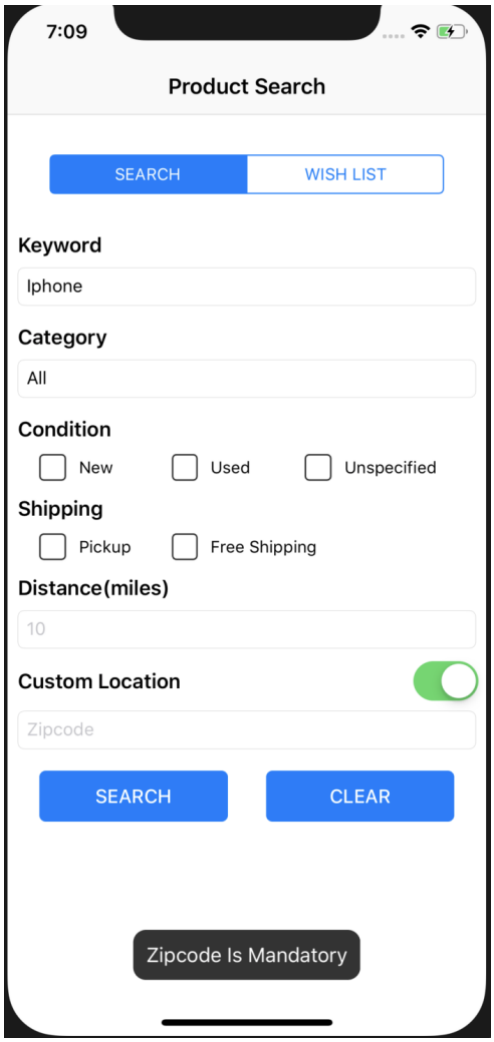


Figure 5: Validation for Zipcode

Once the validation is successful, you should execute an HTTP request to the Node.js script located in the GAE/AWS/Azure back-end, which you should have completed in Homework 8, and then navigate to the Search Results page.

5.2 Search Results

When the user taps the "SEARCH" button, your app should display a *big spinner* (Figure 6) before it's ready to show the results page. Then after it gets data from your backend, it should hide the spinner and display the result page as a UITableView, as shown in Figure 7.

Figure 6: Display a big spinner while searching

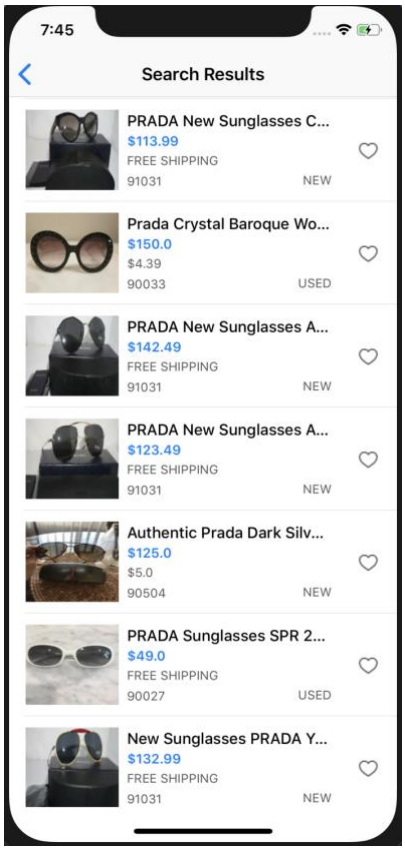


Figure 7: List of search results

Each of the **UITableView's Cell** should have the following:

- Image of the product
- Title of the product
- Price of the product
- Shipping cost of the product: If shipping cost is 0, 'FREE SHIPPING' should be displayed instead.
- Zip code
- Condition of the product (New / Used / Refurbished /NA).
- A heart-shaped "WishList" button

See homework 8 for more details about these fields.

5.2.1 Condition

The condition of the product is shown according to the following mapping:

Condition ID	Condition
1. ‘1000’	NEW
2. ‘2000’ or ‘2500’	REFURBISHED
3. ‘3000’ or ‘4000’ or ‘5000’ or ‘6000‘	USED
4. Any other	NA

5.2.2 Wishlist button

Tapping the Wishlist button (the heart) would add the corresponding product into the Wish list, a message should be displayed at the bottom of the app and the Wishlist button should be filled with red, as shown in Figure 7. Tapping that button again would remove that product from the wish list, and a similar message should also be displayed to indicate that the product has been removed from wish list and the button is changed back to empty button as shown in Figure 8.

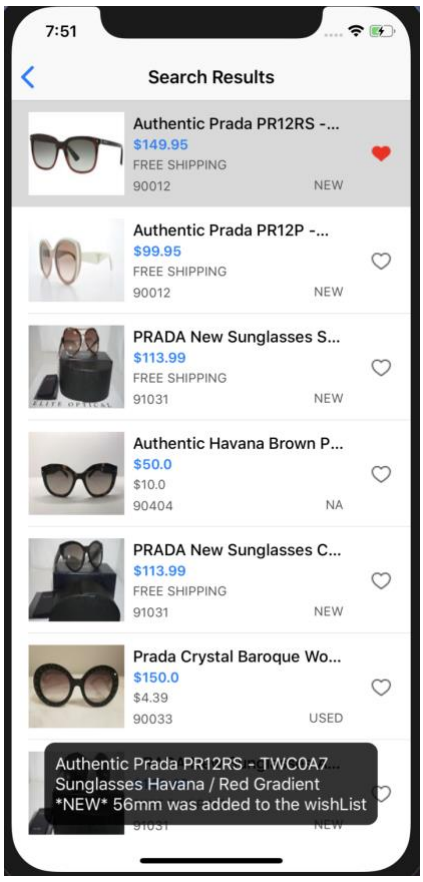


Figure 7: Message on adding to wish List

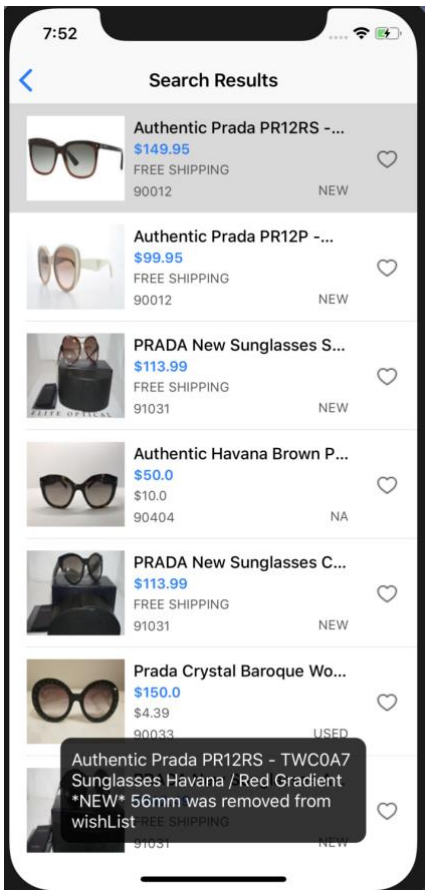


Figure 8: Message shown for removing from Wish List

5.2.2 No Results

When the search query returns zero results or fails to get the results, appropriate message should be displayed as shown in figure 9 and should automatically return to the search form on clicking ‘Ok’.

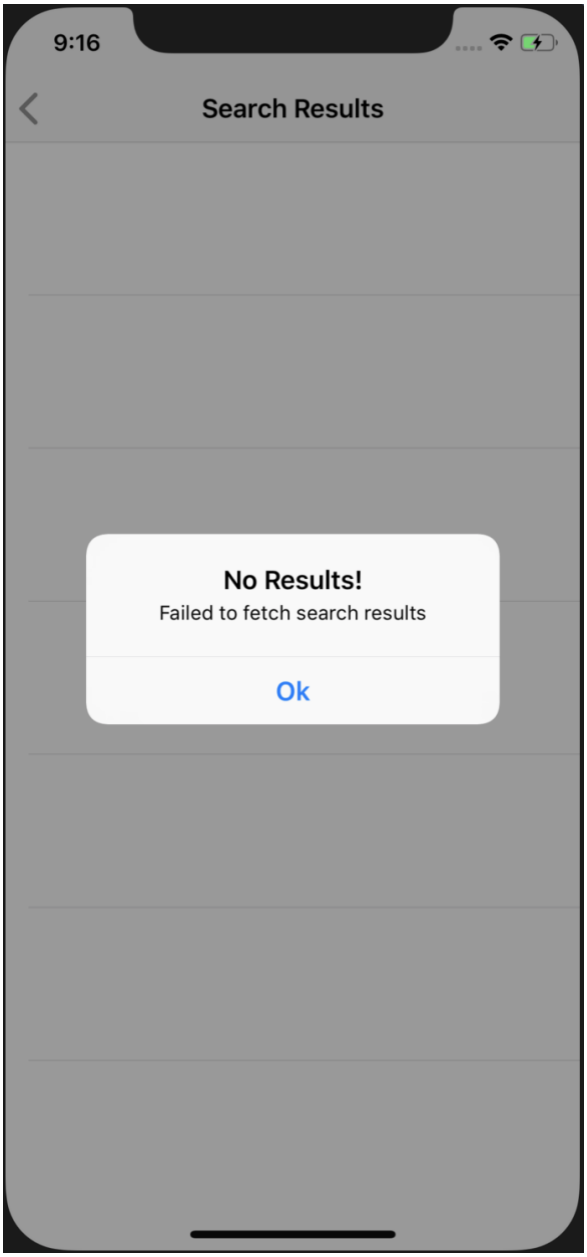


Figure 9: Message on No Results

5.3 Product Details

Tapping on a UITableViewCell in the results table should show details of that product with four tabs: Product Info, Shipping Info, Product photos from google and Similar Products (Figure 10). Note that a spinner should be shown before you are ready to display information in each tab.

All the four tabs share the same **Navigation Bar on the top**. The navigation bar should include the following elements:

- **Back button** which navigates back to the search results list.
- **Share button (Facebook icon**) to share the product details on Facebook. Once the button is tapped, a web page should be opened to allow the user to share the product information on Facebook, as shown in Figure 11. **See homework 8 and homework video for how it works and the format of the Facebook content.**

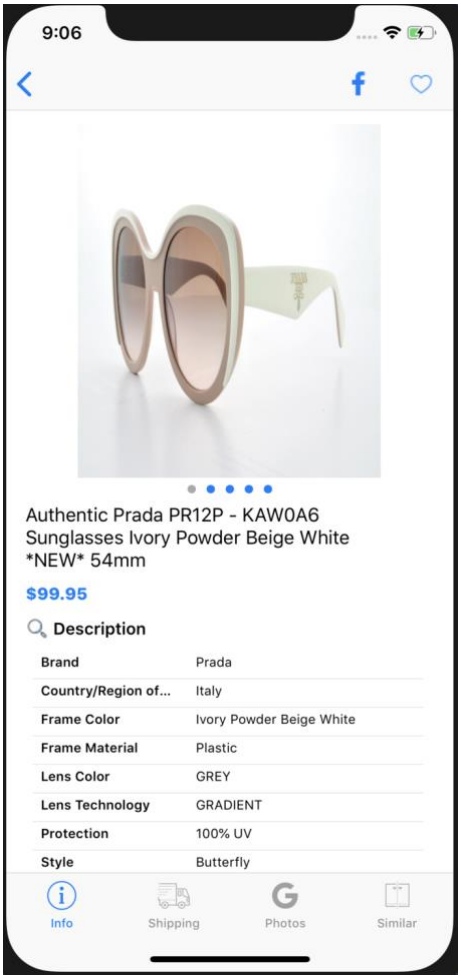


Figure 10: Product Details with 4 tabs

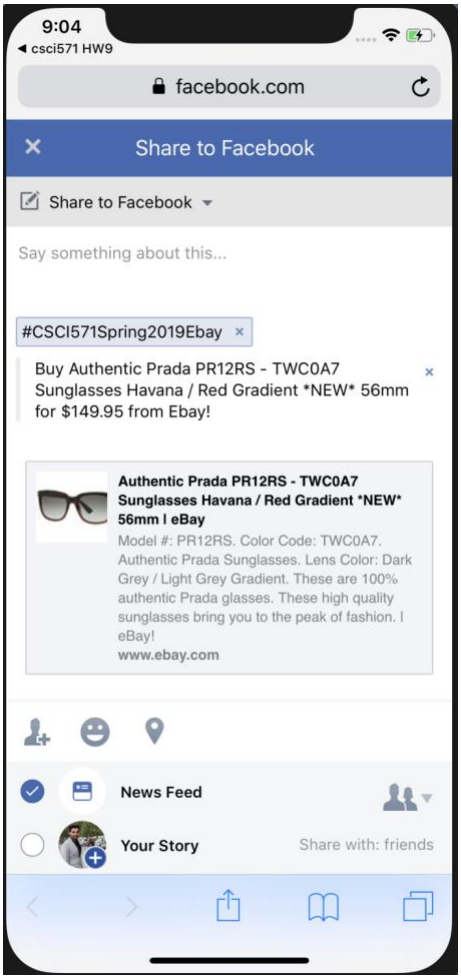


Figure 11: Share Product on Facebook

- **Wishlist button** to add/remove the product to/from the wish list and display an appropriate message at the bottom of the screen (same as 5.2.1). **See video for detailed flow**.

5.3.1 Product Info Tab

The product info tab contains the following:

- A ‘UIScrollView’ and a ‘UIPageControl’ to **horizontally** display the product images. The horizontal scroll is embedded with ‘UIPageControl’ to show the total number of images as well as the current image being viewed.
- The product title using a maximum of three lines.
- Product price.
- Description about the product: All the elements of the ‘NameValueList’ in ‘ItemSpecifics’ property of the ‘Item’ displayed as ‘UITableView’.

  If there are multiple values for a field, Just show the first one.

  See homework 8 for more details about each field.

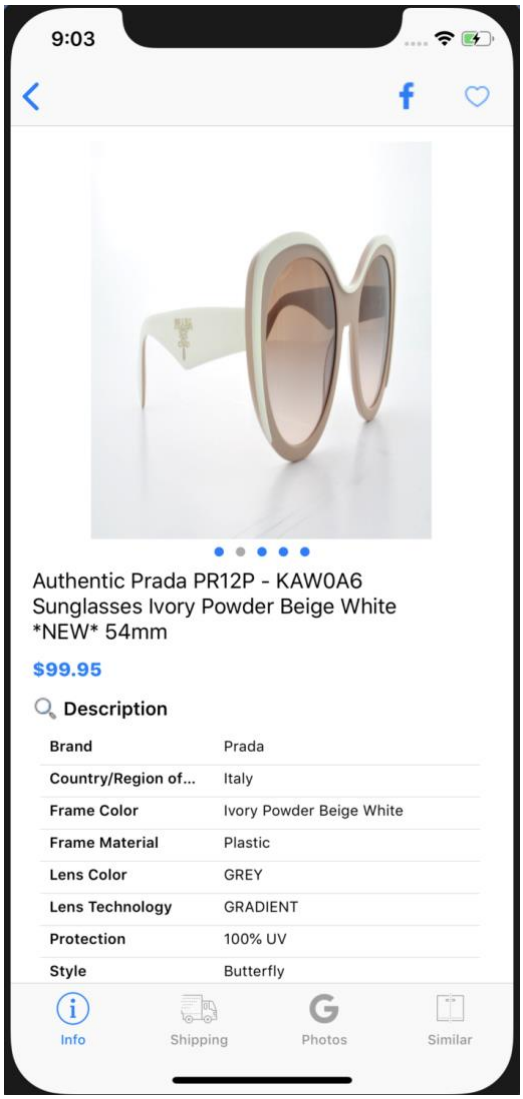


Figure 11: Product Info Tab

5.3.2 Shipping Tab

The Shipping tab is divided vertically into 3 sections as shown in Figure 12:

Section 1: Seller Information

Seller information contains the following:

- Store Name: Hyperlink with the name of the store name linked to the storeURL from the 'StoreName' and 'StoreURL' attributes of 'Storefront' property of Item.
- Feedback Score: 'FeedbackScore' of the 'Seller' attribute.
- Popularity: 'PositiveFeedbackPercent' of the 'Seller' attribute.
- Feedback Star: The color and type of star used for icon depending on the 'FeedbackRatingStar' property of the 'Seller' attribute: 'star' icon for 'shooting' value and 'starborder' icon otherwise.

Section 2: Shipping Information

Shipping information contains the following:

- Shipping Cost: 'FREE' if cost is 0.
- Global Shipping: 'GlobalShipping' property of the 'Item': 'Yes' if the value is 'True' and 'No' if 'False'.
- Handling Time: 'HandlingTime' property of the 'Item'.

Section 3: Return Policy

Return policy section contains the following:

- Policy: 'ReturnsAccepted' attribute of 'ReturnPolicy' key of the 'Item'.
- Refund Mode: 'Refund' attribute of 'ReturnPolicy' key of the 'Item'.
- Return Within: 'ReturnsWithin' attribute of 'ReturnPolicy' key of the 'Item'.
- Shipping Cost Paid By: 'ShippingCostPaidBy' attribute of 'ReturnPolicy' key of the 'Item'

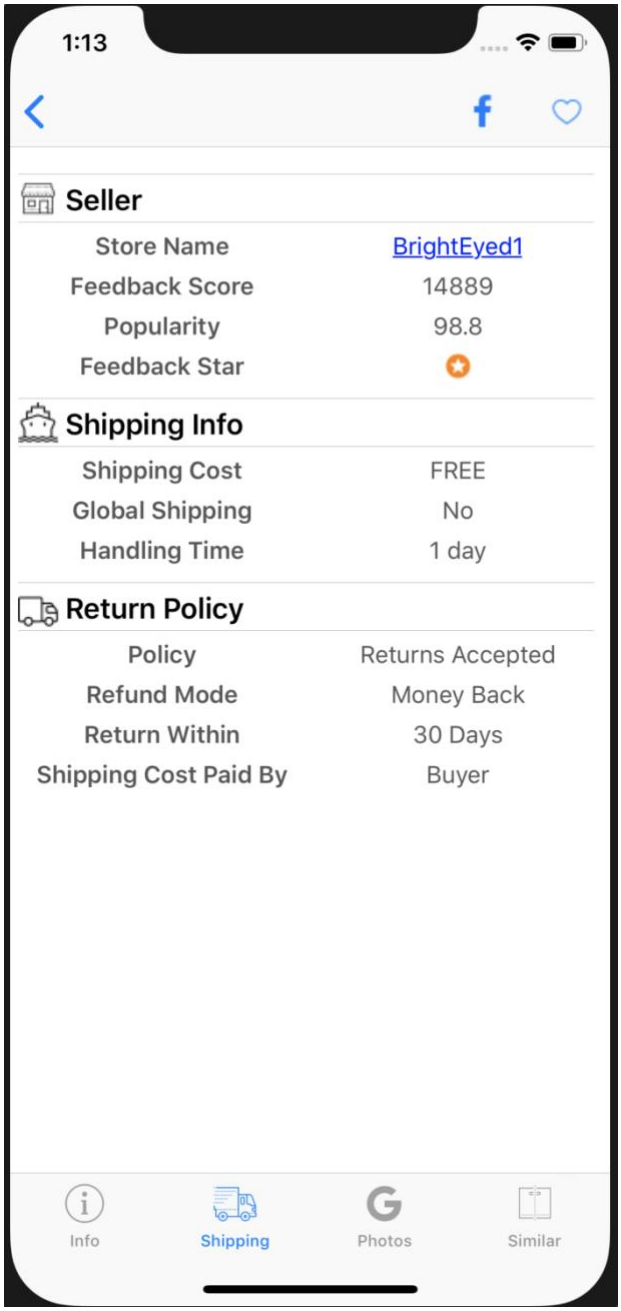


Figure 12: Shipping Tab

Error Handling:

- If any of the above-mentioned fields is missing in the API response, it should NOT be displayed.
- If ALL of the fields of a particular section are missing, the entire section should not be displayed, not even the Header of the section.

5.3.3 Google Photos Tab

Google Images Fetched using the title of the product and shown as 'UIScrollView' with vertical scroll.

See homework 8 for more details of the Google Photos API.

Figure 13: Google Photos Tab

5.3.4 Similar Items Tab

This tab displays the Similar Items of the product, same as homework 8.

As shown in Figure 14, you should use two segmented controls – one to switch between which parameter to sort the Similar Items on and the other to decide in what order to sort it on.

Ascending/Descending option should be disabled when user select “Default

The Items are shown in a “UICollectionView”.

Each of the **UICollectionView’s Cell** should have the following:

- Image of the product
- Title of the product
- Price of the product
- Shipping cost of the product
- Days Left

See homework 8 for more details about these fields.

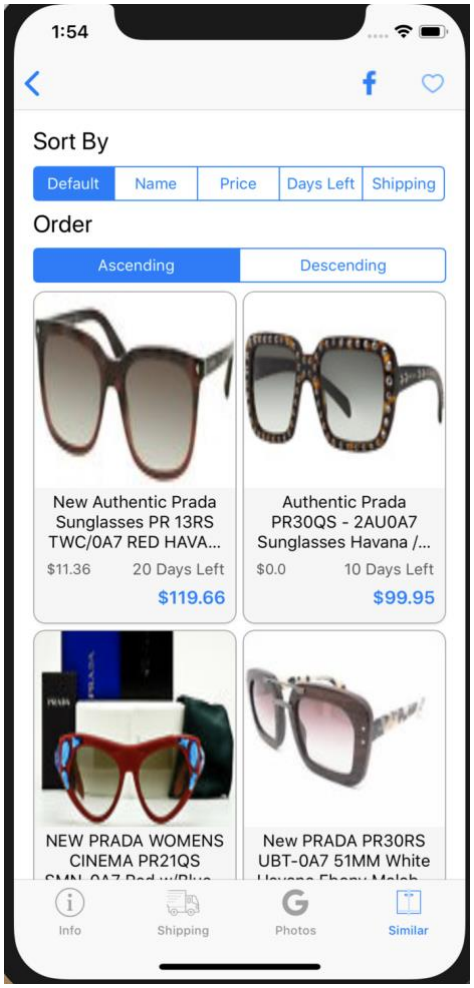


Figure 14: Similar Items Tab

Note that each of the cell can be tapped and then a new Safari instance should show the corresponding product's eBay link as shown in Figure 15.

The ebayLink for the item is the 'viewItemURL' property of 'itemRecommendations' attribute of 'item'.

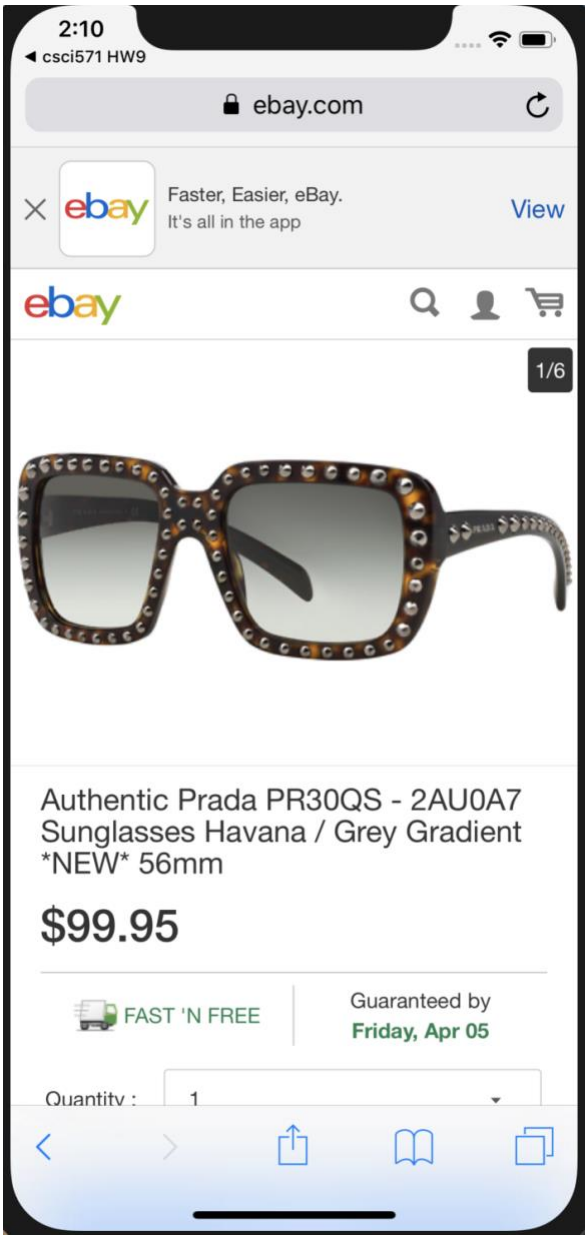


Figure 15: eBay Link

5.4 Wish List

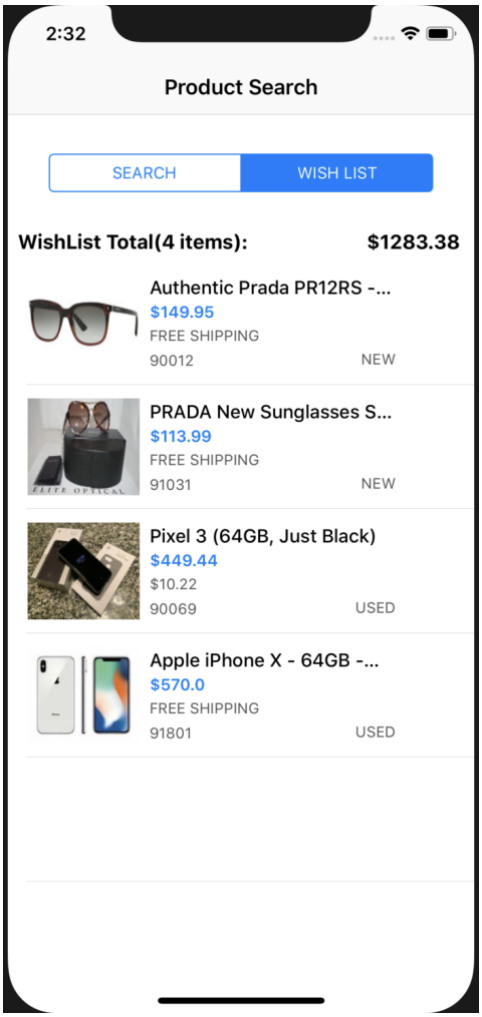
5.4.1 Wish List Implementation

Use a segmented control in the main screen to switch between the search page and the WishList page. The Wish List products should be displayed in a “UITableView”. Each of the “UITableViewCell” has the exact same structure as the search results cell but has **NO wishList button** as show in Figure 16. If there are no items in Wish List, a ‘No Items in Wishlist’ message should be shown (Figure 17).

The total number of items in the wishList and sum of price of all the items should be displayed. (Figure 16)

Tapping on a cell should have the **EXACT** same behavior as tapping on a cell in the search results page.

Use “**UserDefaults**” to store Wish List data.



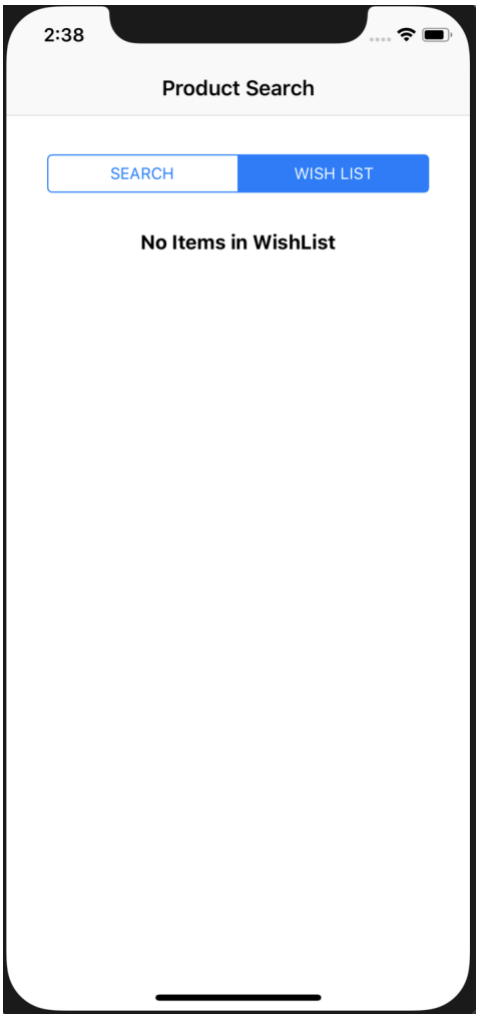


Figure 16: Wish list **Figure 17: No Items in Wish List**

5.4.2 Wish List Delete

The user should also be able to remove a product by left-swiping a cell and clicking the delete option as shown in Figure 18.

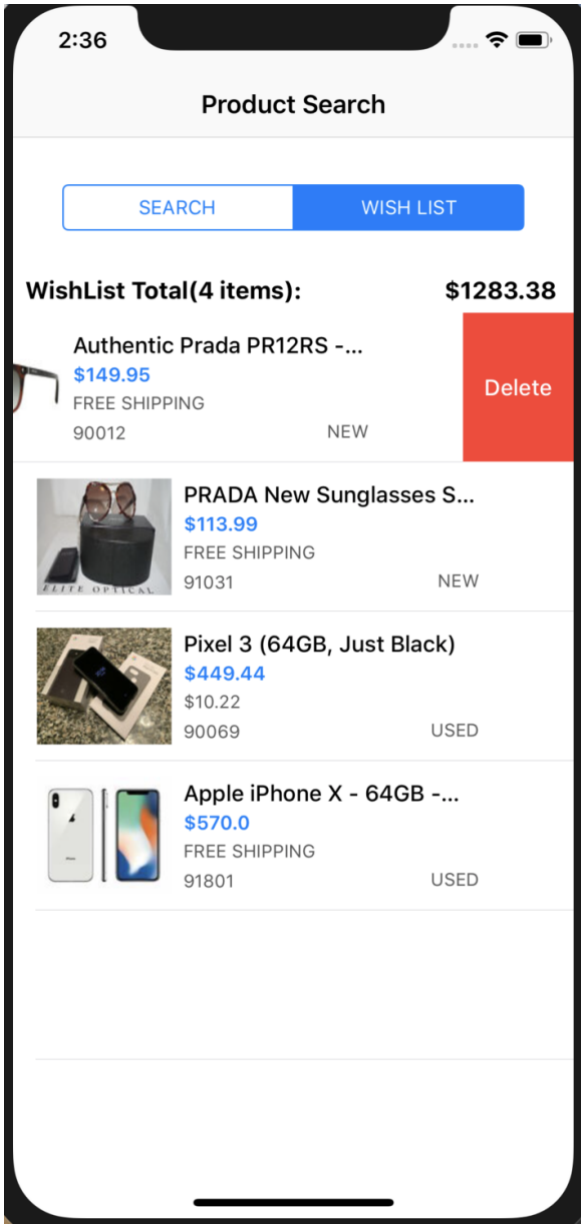


Figure 18: Swipe to remove a product

5.5 Additional Info

For things not specified in the document, grading guideline, or the video, you can make your own decisions. But keep the following points in mind:

- Always display a proper message and don't crash the app, if an error occurs.
- You can only make HTTP requests to your backend (Node.js script on AWS/GAE/Azure)
- All HTTP requests should be asynchronous.

6. Implementation Hints

6.1 Images / Icons

Click to download the images needed for this HW.

Launch Screen

- trojan

Search Form

- checked
- unchecked

WishList Button

- wishListFilled
- wishListEmpty

Product Info Tab

- description

Shipping Info Tab

- Seller
- star
- starBorder
- Shipping Info
- Return Policy

Tab Icons

Info Tab:	icons8-info-80
Shipping Tab:	icons8-in-transit-80
Photos Tab:	icons8-google-96
Similar Tab:	icons8-similar-items-80

Facebook

- facebook

7. Material You Need to Submit

Unlike other exercises, you will have to “demo” your submission “in person” during a special grading session. Details and logistics for the demo will be provided in class, in the Announcement page and in Piazza.

Demo is done on a MacBook using the emulator, and not on a physical mobile/tablet device.

You should also ZIP all your source code (without image files and third-part modules) and push the resulting ZIP file by the end of the demo day to GitHub Classroom.

****IMPORTANT**:**

All videos are part of the homework description. All discussions and explanations in Piazza related to this homework are part of the homework description and will be accounted into grading. So please review all Piazza threads before finishing the assignment.